# Program for Analyzing Knots Represented by Polygonal Paths

**BRETT A. HARRIS, STEPHEN C. HARVEY**

*Department of Biochemistry and Molecular Genetics, University of Alabama at Birmingham, Birmingham, Alabama 35294-0005*

*Received 25 September 1998; accepted 15 January 1999*

**ABSTRACT:** Knots can occur in real and model biological macromolecules. We describe a program for determining whether or not any knots are present in model structures and for classifying those knots that do occur. The program computes the Alexander polynomial, $\Delta(t)$, for any model. This polynomial characterizes the knot in the sense that if two knots have different Alexander polynomials, then the knots are topologically distinct. The Alexander polynomial of the circle, or trivial knot, is $\Delta(t) \equiv 1$. If the computed Alexander polynomial is not identically equal to unity, then the structure is nontrivially knotted, and the program will then determine a lower bound on the minimum number of path crossings. The program is entirely general, and may be used to analyze any closed polygonal path. © 1999 John Wiley & Sons, Inc. J Comput Chem 20: 813–818, 1999

**Keywords:** knots; macromolecular structure; topology; Alexander polynomials; macromolecular modeling

## Introduction

When modeling polymers that are known to be cyclic, it may be desirable to know if the proposed structures are knotted. For example, closed circular DNA molecules may be knotted *in vivo* or *in vitro* by special enzymes (topoisomerases), and if a model closed circular DNA is large enough, it is important to be able to evaluate accurately its state of knottedness. Knots may also occur during the modeling of large RNAs, because basepairing effectively cyclizes large domains. Because there are no RNA structures that are known to be knotted, any model RNA that contains knots should be identified: either the knot is an artifact of the modeling, or it constitutes a prediction whose correctness must be examined experimentally. Similarly, the formation of disulfide bonds in proteins creates topologically closed domains. The existence of any protein knots would be very surprising, so, once again. it would be useful to be

*Correspondence to:* S. C. Harvey; e-mail: harvey@uab.edu
Contract/grant sponsor: National Institutes of Health
Contract/grant number: RR-12255



CCC 0192-8651 / 99 / 080813-06

able to evaluate the state of knottedness of protein structures determined experimentally or put forward as models.

The purpose of this article is to draw attention to a freely available program that gives information about the *knot type* of a cyclic macromolecule. More generally, the program may be used to analyze knots in any closed polygonal path. This program is written in the C programming language, and is an implementation of an algorithm described by Vologodskii et al.[1]

## Theory

A *knot* is defined[2] as the image of a circle under a continuous 1−1 function. Essentially, this means a closed path in Euclidean 3-space that does not intersect itself. In particular, a circle is a knot. Any knot that can be deformed into a circle, without self-intersection, will be called a trivial knot. Note that this definition excludes objects commonly referred to as knots. For example, a ''knot'' tied in a shoestring has free ends, and hence fails to satisfy the mathematical definition. If we were to admit continuous 1−1 paths with free ends, then any two such paths could be deformed into one another without self-intersection. All such knots would be equivalent, resulting in a trivial theory.

Our goal is to determine when two knots are to be considered the same. The knots $K_1$ and $K_2$ and are considered *equivalent* if there is a continuous 1−1 function from Euclidean 3-space onto itself, with a continuous inverse, that maps $K_1$ onto $K_2$. (Note that the requirement that the function be 1-1 would be violated if, during the transformation, a self-intersecting curve were obtained.) We see that the question of whether two knots are equivalent is a topological one. It would be impractical to implement an algorithm based on continuous deformations on a computer for any but the smallest and simplest geometries. However, by using algebraic topology, we may transform the problem into a purely algebraic formulation that lends itself to computational analysis.

There are a number of invariants that can be used to classify knots in such an algebraic formulation, and we have chosen the Alexander polynomial. Other invariants, particularly the Jones or homfly polynomials, are stronger invariants (see, e.g., http://ourworld.compuserve.com/homepages/bob-jenkins/homfly.htm), but their calculation is much more demanding than that of the Alexander polynomial, and they are only suitable for systems that have many fewer crossings than we need to analyze. For a detailed treatment of the theory, see the monograph by Crowell and Fox.[2]

A *knot diagram* is obtained by projecting a knot onto a plane. If the plane of projection changes, then the number of crossings in the projection will usually change as well. Obviously, there is some plane where the number of crossings is minimized. We call this number the minimum crossing number. Knots are tabulated according to their minimum crossing number. In the literature, a collection of all the knots of some minimum crossing number $m$ is ordered according to some (arbitrary) criterion. Then, any knot with $m$ crossings is referred to by the symbol $m_n$, where $n$ is the position of the knot in the list according to the particular order being used.

## *Knot* Program

### IMPLEMENTATION OF ALGORITHM

To simplify the calculations, knots are represented by polygonal paths. In this description, a knot may be described by a sequence of ordered triples representing the three-dimensional coordinates of the vertices of the polygonal path. This is much simpler than having to prepare the data in a ''knot file'' format containing an enumeration of the crossings with directional and crossover parameters, as required by the knot analyzer at the website mentioned earlier.

Supposing the path to have $n$ vertices, numbered 0 through $n - 1$, the program assumes that vertex 0 is connected to vertex 1, vertex 1 is connected to vertex $2, \ldots,$ vertex $n - 2$ is connected to vertex $n - 1$, and finally, vertex $n - 1$ is connected to vertex 0. The choice of vertices will depend on the molecule being investigated. For example, when modeling closed circular DNAs, the vertices might be regularly spaced points along the axis of the double helix, whereas, for polymers such as RNA or proteins, the vertices may be the positions of key backbone atoms such as phosphoruses or alpha carbons.

A challenge in the implementation of the algorithm of Vologodskii et al.[1] is the computation of the determinant of a matrix with polynomial entries. The determinant of a matrix is usually evaluated by transforming the matrix into upper-triangular form—using Gaussian elimination with pivoting, then computing the product of the diagonal

entries. It was necessary to modify this procedure for two reasons.

First, division operations are necessary in the course of Gaussian elimination. We could tackle this problem by using rational functions (pairs of polynomials) for the matrix entries, but this would double the memory requirements for storage of the matrix. A better approach is as follows: Suppose we are given an $n \times n$ matrix, **P**, with entries $p_{ij}$ and that we intend to clear column $i$. Then, the new entry, $\hat{p}_{jk}$, in row $j$ column $k$ is given by:

$$\hat{p}_{jk} = p_{jk} - \frac{p_{ji}\, p_{ik}}{p_{ii}}$$

Multiplying both sides of this equation by $p_{ii}$ yields:

$$q_{jk} \equiv p_{ii}\, \hat{p}_{jk} = p_{ii}\, p_{jk} - p_{ji}\, p_{ik}$$

Next, we replace the entries in row $j$ with $q_{jk}$. Suppose that we have updated $r_i$ rows to clear column $i$. When the matrix is in upper triangular form, the product of the diagonal entries must be divided by $p_{ii}^{r_i}$ to obtain the determinant of the original matrix. This method only requires us to store each $p_{ii}$ along with its corresponding frequency, $r_i$.

Second, for moderately sized test structures (1000 vertices) the number of crossings was as large as 1700. Even though the matrix is sparse, overflow of the coefficients is possible during evaluation of the determinant. The Alexander polynomial, $\Delta(t)$, is unique up to multiplication by $\pm t^m$, where $m$ is any integer.[1] Hence, we may freely multiply or divide any row or column of the matrix by $\pm t^m$ without affecting the value of the determinant. We see that these are ideal pivots, because they do not need to be stored for later division. Beginning with the diagonal entries, a search is made for pivots of the form $\pm 1$ or $\pm t$. Because all of the entries on the diagonal are initially set to one of these values, it is natural to begin searching there. If none are found, the nondiagonal entries are searched. If no entries are of the form $\pm 1$ or $\pm t$, then a pivot is chosen to be a nonzero polynomial of minimal degree, so as to control the degree of the polynomial entries to be computed.

After a row is updated, that row is divided by $t$ if possible. When a column is cleared, each column is divided by $t$, if possible. Then each row is tested for divisibility by the stored factors $p_{ii}$. If the test is affirmative, the operation is performed, and the frequency, $r_i$, is reduced by one. In this manner, coefficient overflow becomes less of a problem than if the divisions were postponed until after computation of the product of the diagonal entries.

In the event that integer overflow occurs, an error message will report the condition. The knot analyzer should be run again using the command line options $-$x and $-$y to rotate the data set. A rotation will affect the number of entries in the matrix, and hopefully alleviate the problem. Of course, it is possible for a knot to be so tangled that the coefficients of the Alexander polynomial exceed the storage for a long integer type. In this case, the only solution is to create an *extra*-long integer type, or to rewrite the program in C++, or some other language that easily handles larger integer types.

## Data Format

The program was developed to work with data files readable by the molecular modeling package YAMMP.[3] These files contain the coordinates of the atoms, organized in a format called an archive. An archive file consists of a file header and one or more records, possibly separated by comments. The file header consists of the string ''ARC3'' followed by a sequence of three integers: *version*, *numatom*, and *filestat*. The parameter *numatom* is the number of atoms or vertices of the knot. The other two parameters are irrelevant for purposes of the knot analyzer. Each record begins with a record header, and is followed by a list of the three-dimensional coordinates of *numatom* vertices arranged in order of traversal. The record header is a sequence of numbers: *creator*, *time*, *numdimen*, and *recstat*. The parameter *time* is a floating point number, and the other three parameters are integers. If desired, comments may be entered between, but not within, records. A comment begins with the ''#'' character, and ends with a carriage return. Comments are ignored by the knot analyzer. Data files may be imported in either text or binary format, and YAMMP utilities can be used for converting between these.

If the user generates models using YAMMP, then *numdimen* must be set equal to three. If compatibility with YAMMP is not required, then the values of all integers and floats in the header and record files may safely be set to zero, with the exception of *numatom*, which must be set to the number of vertices of the knot. Note that if the path intersects

```
ARC3      1000      30        0

# The following path represents the trefoil knot.
# There are thirty vertices (atoms) in the record.
# Beginning Record No.1

10        0         3         0

 30.00      0.00      0.00
 25.66     11.43      5.88
 15.45     17.16      9.51
  5.23     16.08      9.51
 -1.24     11.84      5.88
 -5.00      8.66      0.00
 -9.64      7.00     -5.88
-16.54      3.52     -9.51
-22.59     -4.80     -9.51
-22.73    -16.51     -5.88
-15.00    -25.98      0.00
 -2.94    -27.94      5.88
  7.14    -21.96      9.51
 11.31    -12.57      9.51
 10.88     -4.84      5.88
 10.00      0.00      0.00
 10.88      4.84     -5.88
 11.31     12.57     -9.51
  7.14     21.96     -9.51
 -2.94     27.94     -5.88
-15.00     25.98      0.00
-22.73     16.51      5.88
-22.59      4.80      9.51
-16.54     -3.52      9.51
 -9.64     -7.00      5.88
 -5.00     -8.66      0.00
 -1.24    -11.84     -5.88
  5.23    -16.08     -9.51
 15.45    -17.16     -9.51
 25.66    -11.43     -5.88
```

**FIGURE 1.** Sample data file in YAMMP archive format. The data are for a polygonal path representation of the trefoil or cloverleaf knot.

itself in 3-space, the user will be informed of the error. The data must be modified in this event.

One advantage of using the YAMMP archive format is that these files can also be read by the YAMMP *exam* utility, which can give graphic displays of coordinate sets using RIBBONS.[4]

A typical data file is shown in Figure 1. This 30-sided polygon contains the simplest possible knot, a trefoil. The Alexander polynomial for a trefoil is $t^2 - t + 1$, and it has three crossings (Fig. 2).

## Command Line

To use the knot analyzer, enter the name of the program followed by the name of the archive file containing the data to be analyzed. Several command line options are available, and are summarized in Table I. These options may be entered in any order, and may precede or follow the name of the archive. The user will be informed if an improper value is entered after an option, or if the filename is omitted. If the user simply enters the name of the program, the list of options will be displayed.

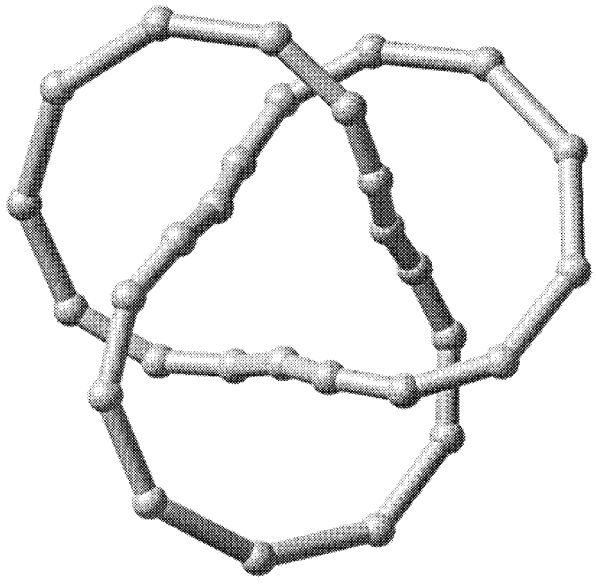

**FIGURE 2.** RIBBONS representation of the trefoil, generated by the YAMMP *exam* utility. The 30 vertices are shown as spheres. Because there are only three crossings, this is a regular projection of the structure.

## Output

The main output of the program, with no options in use, is the Alexander polynomial $\Delta(t)$. If the polynomial is of degree $n$ with coefficients $a_k$, then the output will be in the form $(a_n, a_{n-1}, \ldots, a_1, a_0)$. The Alexander polynomial for the trivial knot is $\Delta(t) \equiv 1$. If the output polynomial is not identically equal to one, then the structure is guaranteed to be nontrivially knotted. However, if the output polynomial is identically equal to one, the test is indeterminate, although the structure is probably unknotted. For more information about this latter difficulty, see the discussion by Vologodskii et al.[1]

In addition to computing the Alexander polynomial, a lower bound is computed for the minimum number of crossings in a regular projection of an equivalent knot. The file, *knot.db*, contains a list of the Alexander polynomials for all knots with 11 or fewer crossings. These polynomials were obtained

**TABLE I.**
**List of Command Line Options.**

| | |
|---|---|
| `-c` | Prints a countdown of the number of rows remaining in the matrix during computation of the determinant. |
| `-f` | Followed by a positive integer, indicates that this is the first vertex (atom) in the record to be read. The default value is one. |
| `-l` | Followed by a positive integer, indicates that this is the last vertex (atom) in the record to be read. The default value is *numatom*, listed in the file header. |
| `-r` | Followed by a positive integer, selects that record number from the archive file. The default value is one. |
| `-s` | Print the size of the Alexander matrix. |
| `-t` | Print the amount of time (seconds) it took to compute the determinant. |
| `-x` | Followed by a floating point number, rotates the data set about the *x*-axis by the supplied number of degrees. |
| `-y` | Followed by a floating point number, rotates the data set about the *y*-axis by the supplied number of degrees. |
| `-dx` | Followed by a floating point number, sets the step size by which the data is rotated about the *x*-axis when searching for a regular projection. The default value is $10^{-5}$. |
| `-dy` | Followed by a floating point number, sets the step size by which the data is rotated about the *y*-axis when searching for a regular projection. The default value is $10^{-5}$. |

from the website, http://www.ifh.de/~aneziris/results.html, which is maintained by Dr. Charilaos Aneziris, who has kindly granted us permission to redistribute the table. They are ''normalized'' as suggested earlier,[1] by requiring that $\Delta(0) > 0$. To get the lower bound, and thereby estimate the complexity of the knot, the Alexander polynomial of the knot is compared with polynomials in the list given in *knot.db* in increasing order of minimal number of crossings. Upon the first occurrence of a match, the number of crossings for the corresponding knot is recorded. If no match is found, then the knot must be equivalent to a knot with at least 12 crossings.

If two knots are joined by splicing them together, the Alexander polynomial of the new knot is equal to the product of the Alexander polynomials of the constituent knots. In such a case, the knot is composite, the computed Alexander polynomial is factored over the list of polynomials, and the corresponding numbers of crossings are summed and recorded. If the remainder after division is not in the list, then, under the assumption that the knot is composite, the component knot whose Alexander polynomial is equal to the remainder, must itself must be equivalent to a knot with at least 12 crossings.

The minimum of all these recorded numbers must be a lower bound of the minimum crossing number of the structure. As an example, analysis of the trefoil of Figures 1 and 2 is shown in Fig. 3.

```
prompt 1% knot -s -t -c 3_1.k
matrix size is: 3 x 3
position = 2
determinant took 0.00 seconds to compute
Alexander polynomial is: (1, -1, 1)
equivalent to a knot with at least 3 crossings
```

**FIGURE 3.** Knot analysis for the data given in Figure 1. Input is shown on the prompt line, and output on the subsequent lines. The Alexander polynomial for a trefoil is $t^2 - t + 1$.

## Where to Obtain *Knot*

The knot analyzer is available at http://uracil.cmc.uab.edu/Publications. The program is written in C, and consists of three files, *knot.h*, *knot.c*, and *knot.db*. The header file, *knot.h*, contains structure definitions, macros, global variable declarations, and function prototypes. The file *knot.c* contains the source code of the knot analyzer. Finally, *knot.db* is a list of the Alexander polynomials of knots with 11 or fewer crossings. After compilation, the object file must be linked with the standard math library.

## Discussion

Topology can be an important feature of any model of a biopolymer, whether the state of knottedness is the subject of the investigation (as might occur in closed circular DNAs), or whether one wants to be certain that artifactual knotting has not occurred during modeling. We are currently using this program to evaluate knots in model closed circular DNAs in viral capsids and artifactual knotting in models of ribosomal RNAs developed in our laboratory and elsewhere.[5] (Although no naturally occurring knots have yet been observed in RNAs, an RNA topoisomerase has been discovered,[6] and knots have been reported in protein

structures.[7,8] The *knot* program will allow the detection of knots in model structures, so a decision can be made as to whether the knot is an artifact of the modeling, or a concrete and experimentally testable prediction.)

The program described here allows for rigorous quantitative evaluation of the state of knottedness by evaluating the Alexander polynomial. Although this polynomial is not as strong an invariant as the Jones or homfly polynomial, it can be evaluated much more quickly than those invariants, permitting the examination of very large systems such as macromolecules, and it is the method of choice when one simply needs to know whether or not the structure is knotted. The very simple format of the input file (sets of triples corresponding to the coordinates of successive points along the polymer backbone) is another advantage of this approach.

The criterion for determining if a structure is trivially knotted is simple: If $\Delta(t) \equiv 1$, we can be reasonably sure that the model is trivially knotted. Any other Alexander polynomial corresponds to a model that is nontrivially knotted. Furthermore, the computation of the Alexander polynomials is fast: average running time for cyclic models with 500 to 800 vertices is about 3 minutes on a Silicon Graphics Octane, and we have evaluated systems containing over 500 crossings in less than 10 minutes. The knot analyzer provides a quick way to determine if a cyclic structure is nontrivially knotted.

## Acknowledgments

This work was inspired by questions raised by Javier Arsuaga (Florida State University) and by an anonymous questioner at the 1998 RNA Society meeting. We are indebed to Dr. Charilaos Aneziris for allowing us to redistribute data from his website. We thank Robert Tan and Margaret VanLoock for technical assistance and stimulating discussions.

## References

1. Vologodskii, A. V.; Lukashin, A. V.; Frank-Kamenetskii, M. D.; Anshelevich, V. V. Sov Phys-JETP 1974, 39, 1059.
2. Crowell, R. H.; Fox, R. H. Introduction to Knot Theory; Ginn and Co.: Boston, 1963.
3. Tan, R. K.-Z.; Harvey, S. C. J Comput Chem 1993, 14, 455.
4. Carson, M. J Appl Crystallogr 1991, 24, 958.
5. VanLoock, M. S.; Harris, B. A.; Harvey, S. C. J Biomolec Struct Dynam 1998, 16, 709.
6. Wang, H.; DiGate, R. J.; Seeman, N. C. Proc Natl Acad Sci USA 1996, 93, 9477.
7. Liang, C.; Mislow, K. J Am Chem Soc 1995, 117, 4201.
8. Takusagawa, F.; Kamitori, S. J Am Chem Soc 1996, 118, 8945–8946.